FINAL PROJECT SUBMISSION REPORT

ON

**Plural Formation of Telugu Nouns Using Morphology in Prolog**

Submitted by

**M.Durga Prasad**

( MCL-2/H-012/2014 )

Course Project Submitted in **PROLOG**

**MASTER OF COMPUTATIONAL LINGUISTICS**

COURSE INSTRUCTOR

Prof. **Dr. M. HARI PRASAD**

****

**The English and Foreign Languages University**

Plural Formation of Telugu Nouns using

Morphology in Prolog

***Abstract*** *—* Telugu is an Indian language spoken byover 50 million people in the country. The language is rich in literature and has been studied by native and foreign linguists significantly, yet it has not benefited significantly from the recent advances in computational approaches for linguistic or statistical processing of natural language texts. However with the recent progress in standardization of machine representation of text, applications like machine translation and information retrieval are beginning to surface, for example with the collaborative efforts under the umbrella of Digital Library of India (DLI). There is a need for a morphological generator for Telugu that forms an integral part of applications like machine translation and universal dictionary.

Here I am present the development of a tool, called Plural formation of Telugu Nouns using Morphology,that can generate plural forms of nouns of Telugu. It has been developed based on the previously established linguistic analyses of Telugu by C.P. Brown and H. Krishnamurthy. The tool is developed in Prolog.

**Plural Formation of Telugu Nouns**

Last syllable for plural nouns is always the *plural suffix* lu, although plural formation happens in a number of different ways. The regular way of form-ing the *nominative plural* of a common noun is to add the plural suffix lu to the *nominative singular*. For ex-ample:

aavu aavulu

anna annalu

kurchii kurchiilu

pette pettelu

However, on addition of the plural suffix, there may be a *san’dhi*(conjugation) formation, because of whichlubecomes l’u or l’l’u or n’d’lu. There is also a list of nouns that do not form plural according to these rules. Plurals sometimes have variant forms in use (for example kannulu or kal’l’u).

*Rules of san’dhi formation*

Let nominative singular be referred to as *stem* in the context of plural formation

1. If stem final is: [t’/n’t’/n’d’] + [i/u], then the final vowel [i/u] is lost before the plural suffix lu.

[t’/n’t’/n’d’] + [i/u] [t’/n’t’/n’d’]

+ lu

t’it’lun’t’in’t’lu

t’it’lun’t’i

n’t’lu

n’d’in’d’lut’u

t’lun’t’un’t’lu

n’d’un’d’lun’d’lu

freely becomesl’l’u

1. If stem final is: [d’i/d’u/lu/ru] or if stem is more than 2 syllables and ends in [li/ri], the final syllable becomes [l’] before adding [l’u].

*Exception:* Masculine nouns of Sanskrit originending in d’u replace d’u by lu

Example: sneihitud’u sneihitulu

1. Stem final [t’t’/d’d’]+[i/u] becomes [t’/d’]+l’u
2. Stem final [llu/nnu] becomes n’d’lu or l’l’u.

*Exception****:***Following stems addluto the basicstem to form plural: Pannu, vennu, ponnu,jannu, tannu, t’annu.

1. Stem final [an’/aan’] is replaced by aa and stem final [en’] by e’ before plural suffix lu.
2. Stem final aayi, having more than 2 syllables add lu.
3. Stem final [y/yy]+i is replaced by tulu. Only 3 nouns in this class: cheyyi, goyyi, neyyi.
4. If above rules do not apply and stem ends in i, then
5. If stem is 2 syllables, or if 3 syllables and middle vowel is other than i, then i changes to u before lu. If middle syllable is i, then that also changes to u, unless the noun is of Sanskrit origin. (atithi, parithi, samiti).

***Exceptions to plural formation****:*

The following nouns to do not conform to plural formation rules given above:

raayi raal’lu

poyyi poyyilu

pen’d’li pen’d’in’dd’lu

pel’l’i pelli l’l’u

vari vad’lu

+gaaru +gaarlu

eddu eddulu ed’lu

veyyi veilu

+saari +saarlu

cheinu cheilu

peinu peilu

eid’u ein’d’lu ei’l’lu

+gaad’u +gaal’l’u

allud’u allun’d’lu allul’l’u

manamaraalu manamaraan’dlu manamaraal’l’u

Plural generation in My program is performedbothbased on Brown’s informal description and Krishna-murthy’s linguistic formalism described in [7, 8]. However, the latter generation although very reliable in most classes, is erroneous for some (one or two) exceptions.

**Programing Approach**: In the prolog programing approach of the Telugu Nouns can be declared in 7 types based on suffixes. It shown in below:

## Type 1: NOUN ends with the letter "A"

|  |  |
| --- | --- |
| **Noun** | **Plural form** |
| Ann’a | Anna’lu |
| Naann’a | Naanna’lu |
| Karr’a | Karra’lu |

## Type2: NOUN ends with the letter "E"

|  |  |
| --- | --- |
| **Noun** | **Plural form** |
| Pett’e | pette’lu |
| Katt’e | katte’lu |
| Midd’e | midde’lu |

## Type 3: NOUN ends with the letter "AM"

|  |  |
| --- | --- |
| **Noun** | **Plural form** |
| Lanch’am | Lanch’alu |
| Add’am | add’alu |
| Randr’am | randr’alu |

## Type 4: NOUN ends with the letter "I"

|  |  |
| --- | --- |
| **Noun** | **Plural form** |
| samit’i | Samit’ulu |
| Pand’i | pand’ulu |
| Atith’i | atith’ulu |

## Type 5: NOUN ends with the letter "U"

|  |  |
| --- | --- |
| **Noun** | **Plural form** |
| Vill’u | Villu’lu |
| goov’u | goovu’lu |
| Ill’u | illu’lu |

## Type 6: NOUN ends with the letter "MU"

|  |  |
| --- | --- |
| **Noun** | **Plural form** |
| Varam’u | varamu’lu |
| penam’u | penemu’lu |
| kancham’u | kanchamul’u |

## Type 7: NOUN ends with the letter "Du"

|  |  |
| --- | --- |
| **Noun** | **Plural form** |
| Ramu’Du | Ramu’llu |
| Allu’Du | Allu’llu |
| Devu’Du | devu’llu |

**Programming design**: A simple programming logic of prolog is writing the choices of the program and declaring the noun plural forms with RTN as shown in below. If you enter go in prolog listener it displays the choices and your choice is yes it goes further step that is reading the word class and declaring the class of the variable for suffix addition and deletion in the RTN’s. Otherwise it stop the excution process.

**Representing RTNs in Prolog**: Since an RTN consists of a set of networks, as opposed to the single monolithic FSTN, the data structure we use to represent the RTN needs to have a way of indicating which bits belong to which subnetwork. If we think of each subnetwork having a name, then we can add a further argument to each predicate of our existing FSTN data structure and use this argument to indicate which component of the network the clause belongs to. This leads to a predicate of the form:

go:-

write('PROGRAM FOR PLURALS FORMATION OF TELUGU NOUNS'),

nl,

write('If your NOUN ends with the letter "A" (like amma), enter 1:'),

nl,

write('If your NOUN ends with the letter "E" like(pette,katte), enter 2:'),

nl,

write('If your NOUN ends with the letter "AM" (like shapam), enter 3:'),

nl,

write('If your NOUN ends with the letter "I",(like rayi,pandi), enter 4:'),

nl,

write('If your NOUN ends with the letter "U",(like raju,goovu), enter 5:'),

nl,

write('If your NOUN ends with the letter "MU",(like varamu), enter 6:'),

nl,

write('If your NOUN ends with the letter "udu"/olu/ili , enter 7:'),

nl,

write('Please Enter your choice NUMBER:'),

read(Class),

plural(Noun,Nounplu,Class).

go(yes):-go.

go(no):-write(' THANK YOU').

In this using RT networks the lexicon class, noun and noun attached to the next network it was using lists and concatenation of the lists are add or delete the characters at end of the root in stack. Finally the stack exhibits the output.

A list of terms can be represented between brackets .As a special form of direct pattern matching, [H|T] matches any list with at least one element:

H matches the head of the list,

T matches the tail.

Using these pattern matching facilities, values can be specified as the

intersection of constraints instead of by direct assignment.

Lists are ordinary structures with syntactic sugar added.

The notation abbreviates terms constructed with the predefined

“.” function symbol and the special atom [].The list processing is shown in below:

plural(Noun,Nounplu,class):-

write(' Enter a TELUGU noun which ends with the letter "\_":-'),

nl,

read(Noun),

name(Noun,Y),

dummy\_stack(Y,X),

concat(X,[ ],Nounplu),

write('THE PLURALS FORMATION OF TELUGU NOUN:'),

atomlist\_concat(Nounplu,Z),

write(Z),

write(.),nl,

write('Do u want to enter one more TELUGU NOUN? (yes/no):'),nl,

read(P),go(P).

dummy\_stack([],[]).

dummy\_stack([Head|Tail],[Head1|Tail1]):-

dummy\_stack(Tail,Tail1),

name(Head1,[Head]).

concat([],List,List).

concat([],List,List).

concat([Head|List1],List2,[Head|List3]):-

concat(List1,List2,List3).

Suppose Enter a TELUGU noun which ends with the letter "u" and change the suffix what suffix do you want to add in dammy stack that is concat(X,[ l,u],Nounplu). Then the Amzi! Prolog listener shows the output shown in below:

?- consult('C:\\Users\\cllab\\Desktop\\dp\\finalcode.pro')

yes

?- go.

PROGRAM FOR PLURALS FORMATION OF TELUGU NOUNS

If your NOUN ends with the letter "A" (like amma), enter 1:

If your NOUN ends with the letter "E" like(pette,katte), enter 2:

If your NOUN ends with the letter "AM" (like papam,shapam), enter 3:

If your NOUN ends with the letter "I",(like rayi,pandi), enter 4:

If your NOUN ends with the letter "U",(like raju,goovu), enter 5:

If your NOUN ends with the letter "MU",(like varamu,penamu), enter 6:

If your NOUN ends with the letter "udu"/olu/ili , enter 7:

Please Enter your choice NUMBER:1.

Enter a TELUGU noun which ends with the letter "A":-

amma.

THE PLURALS FORMATION OF TELUGU NOUN:ammalu.

Do u want to enter one more TELUGU NOUN? (yes/no):

no.

THANK YOU

yes

?-

**Result** :

The Noun plural forms are generating using morphological generator. If we give any root noun in Telugu then it will give plural form of the noun. And the program produce the accurate results.

**Limitations** :

1. The program produce the accurate results but it based on only 7 lexicon rules.
2. The plural formation is done by identifying the noun class only.

**Conclusion**:

The noun plurals generates morphological forms of the main lexical classe: noun of Telugu lan-guage. Except for a few irregular classes of nouns, some of which have only a few words per class, the process is automatic. For these exceptions, the user needs to identify the class to which the noun belongs, based on what form it takes for genitive case or based on how its plural is formed. Plural generation is done in two ways: based on informal rules after identifying the class of the noun, or through formal descriptive rules without the requirement for identification of class in which case no user interaction is required. For the latter however, there are 2 syllable endings, for which the generated plural is incorrect.

**References**:

1. Linguistic analyses of Telugu by C.P.Brown and H. Krishnamurthy.
2. Natural Language Processing Techniques in Prolog by Patrick Blackburn and Kristna striegnitz.